R sample codes
Codes written in R to solve real time problems in PhD project

List of codes:

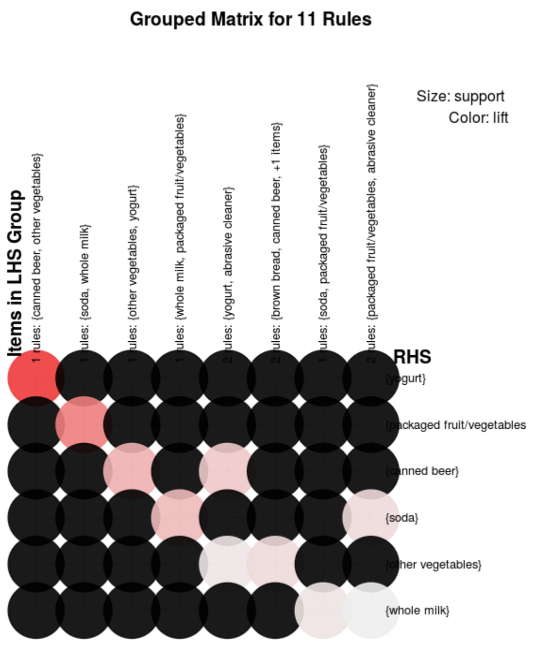
Code-1: Basket analysis in R
Code 2: Heatmap in R
Code-3 : A code for PCA analysis in R
Code-4 : A code for PcoA analysis in R
Code-5 : Volcano plot
Code-6 : Differential gene expression using DEseq2
Code 7 : Survival analysis
Code 8 : Regression analysis
Code 9: Logistic regression
Code 10 : Random Forest
Code 11 : Neural Networks
Code 12 : Waterfall plot
Code 13 : Lollipop plot
Code 14 : K-means clustering in R

Other codes:
WGCNA analysis in R

Code-1: Basket analysis in R (Apriori algorithm):

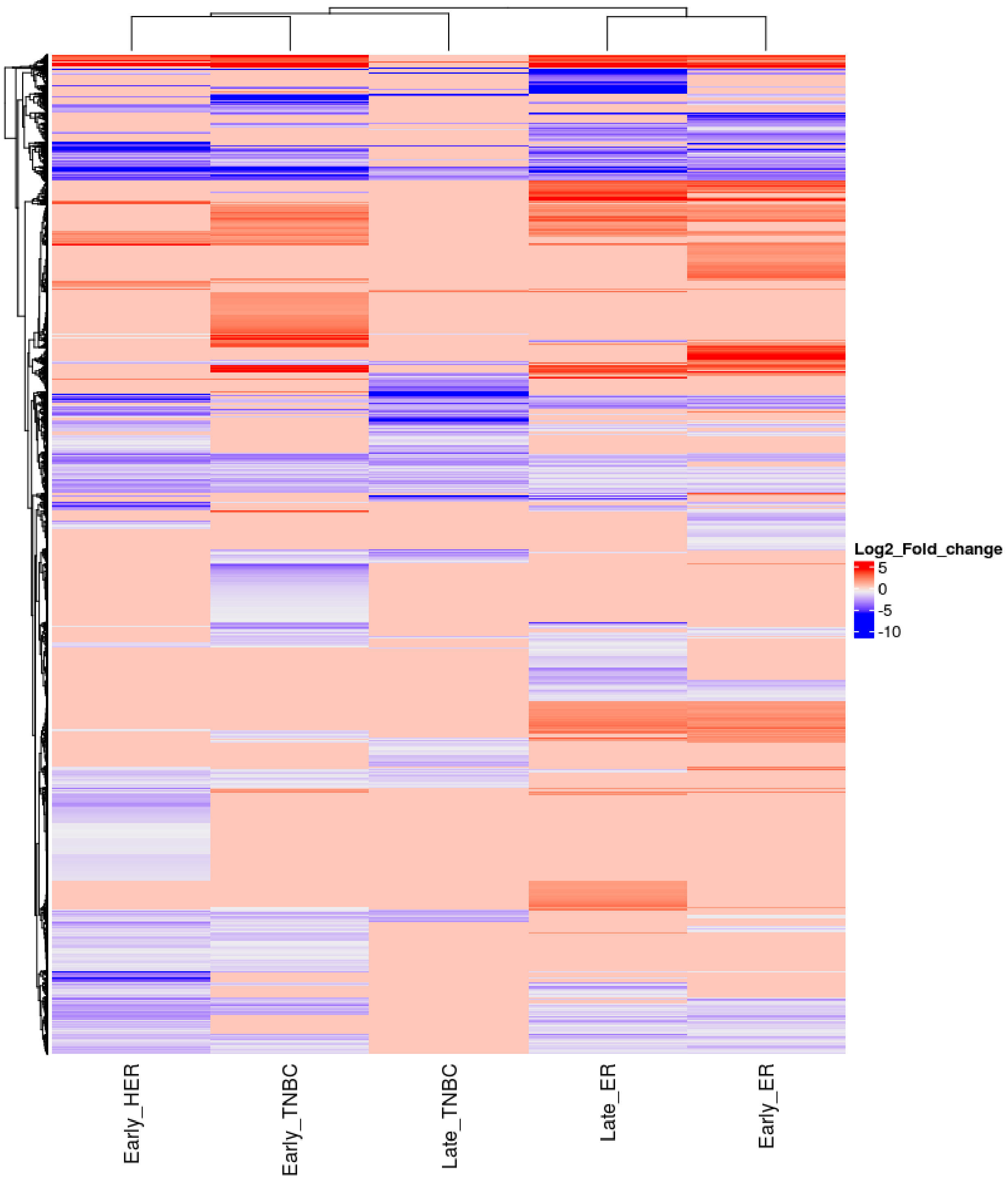
```
getwd()
setwd("/home/ravi/Documents/basket_analysis/pratima_Basket_analysis/")
#df_groceries <- read.csv("selected_mutation_data_patients_3_2.csv")
df_groceries <- read.csv("uniq_out_hnscc_basket_dead.csv")
str(df_groceries)
df_sorted <- df_groceries[order(df_groceries$Member_number),]
df_sorted$Member_number <- as.numeric(df_sorted$Member_number)
df_sorted$itemDescription <- as.factor(df_sorted$itemDescription)
str(df_sorted)
if(sessionInfo()['basePkgs']=="dplyr" | sessionInfo()['otherPkgs']=="dplyr"){
  detach(package:dplyr, unload=TRUE)
}
library(plyr)
df_itemList <- ddply(df_groceries,("Member_number"),function(df1)paste(df1$itemDescription,
collapse = ","))
df_itemList
df_itemList$Member_number <- NULL
##df_itemList$Date <- NULL
colnames(df_itemList) <- c("itemList")
write.csv(df_itemList,"Itemlist_dead.csv", quote = FALSE, row.names = TRUE)
```

```
library(arules)
txn = read.transactions(file="Itemlist_dead.csv", rm.duplicates= FALSE,
format="basket",sep=",",cols=1);
txn
txn@itemInfo$labels <- gsub("\"","",txn@itemInfo$labels)
basket_rules <- apriori(txn,parameter = list(sup = 0.03, conf = 0.6,target="rules",minlen = 1,
maxtime =10000, maxlen =100));
if(sessionInfo()['basePkgs']=="tm" | sessionInfo()['otherPkgs']=="tm"){
  detach(package:tm, unload=TRUE)
}
inspect(basket_rules)
df_basket <- as(basket_rules,"data.frame")
View(df_basket)
write.csv(df_basket, "result.csv")
itemFrequencyPlot(txn, topN = 10)
library(arulesViz)
plot(basket_rules)
plot(basket_rules, method = "grouped", control = list(k = 10))
plot(basket_rules, method="graph", control=list(type="items"))
plot(basket_rules, method = "graph", interactive = TRUE, shading = NA)
#itemFrequencyPlot(txn, support = 0.05)
itemFrequencyPlot(txn, topN = 10)
itemFrequency(txn[, 1:3])
##a visualization of the sparse matrix for the first five transactions
image(txn[1:5])
##visualization of a random sample of 100 transactions
image(sample(txn, 100))
```



<u>Code 2: Heatmap in R:</u>

```
library(pheatmap)
library("RColorBrewer")
setwd("/home/ravi/Documents/R_heatmap/complexheatmap/Heatmap_all_classes/")
test<-read.csv("all_DE_2.csv",sep="\t", row.names=1)
test
test_ref<-read.csv("all_DE_2.csv",sep="\t")


#y<-data.matrix(test)
#pheatmap(test)
#pheatmap(test, scale = "row", clustering_distance_rows = "correlation")
#pheatmap(test, scale = "row", color = colorRampPalette(c("navy", "white", "firebrick3"))(50))
#pheatmap(test, scale = "row", color = colorRampPalette(c("navy", "white",
"firebrick3"))(50),fontsize_row = 7.5, fontsize_col = 15)


library(ComplexHeatmap)
library(circlize)
library(colorspace)
library(GetoptLong)
Heatmap(test)

Heatmap(test,show_row_names = FALSE, km = 15)
Heatmap(test, row_km = 2, column_km = 3, border = TRUE)
Heatmap(test, rect_gp = gpar(col = "white", lwd = 2))


kclus <- kmeans(test,15)
kclus$cluster
table(test_ref$Gene,kclus$cluster)

split <- paste0("Cl\n", kclus$cluster)
default.hmap <- Heatmap(test, split=split)
default.hmap
split <- factor(paste0("Cluster\n", kclus$cluster),
levels=c("Cluster\n2","Cluster\n1","Cluster\n3","Cluster\n4","Cluster\n5"))
reorder.hmap <- Heatmap(test, split=split)
```



Log2_Fold_change
5
0
-5
-10
Early_HER
Early_TNBC
Late_TNBC
Late_ER
Early_ER

Code-3 : A code for PCA analysis in R

```
data.matrix <- read.csv("example.csv", sep = ",", row.names = 1)
head(data.matrix)
pca <- prcomp(data1_BRCA, scale=TRUE)
plot(pca$x[,1], pca$x[,2])
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
barplot(pca.var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent Variation")
library(ggplot2)
pca.data <- data.frame(Sample=rownames(pca$x),
                       X=pca$x[,1],
                       Y=pca$x[,2])
pca.data

ggplot(data=pca.data, aes(x=X, y=Y, label=Sample)) +
  geom_text() +
  xlab(paste("PC1 - ", pca.var.per[1], "%", sep="")) +
  ylab(paste("PC2 - ", pca.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("My PCA Graph")
loading_scores <- pca$rotation[,1]
gene_scores <- abs(loading_scores) ## get the magnitudes
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)
top_10_genes <- names(gene_score_ranked[1:2])
pca$rotation[top_10_genes,1] ## show the scores (and +/- sign)


#remove zero variance columns from the dataset
#data_tsne1 <- data_tsne[ , apply(data_tsne, 2, var) != 0]

dim(data_BRCA_cod)
pca <- prcomp(data_BRCA_cod[1:1205,1:11014], scale=TRUE)

#data_see <- pca[ , apply(pca, 2, var) != 0]
#pca <- prcomp(data_see, scale=TRUE)


pca <- prcomp(data_tsne1, scale=TRUE)
plot(pca$x[,1], pca$x[,2])
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
barplot(pca.var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent Variation")
library(ggplot2)
```

```
pca.data <- data.frame(Sample=rownames(pca$x),
                       X=pca$x[,1],
                       Y=pca$x[,2])
pca.data

ggplot(data=pca.data, aes(x=X, y=Y, label=Sample)) +
  geom_text() +
  xlab(paste("PC1 - ", pca.var.per[1], "%", sep="")) +
  ylab(paste("PC2 - ", pca.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("My PCA Graph")
loading_scores <- pca$rotation[,1]
gene_scores <- abs(loading_scores) ## get the magnitudes
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)
top_10_genes <- names(gene_score_ranked[1:2])
pca$rotation[top_10_genes,1] ## show the scores (and +/- sign)
```

Code-4 : A code for PcoA analysis in R

```
Sel_Genes <- read.csv("example.csv", sep = ",", row.names = 1)
head(Sel_Genes)
Bray_distances <-vegdist(Sel_Genes[,1:100], "bray")
Bray_pcoa <-pcoa(Bray_distances)
Bray_pcoa$values[1:2,]
mds.var.per =
round(Bray_pcoa$values$Eigenvalues/sum(Bray_pcoa$values$Eigenvalues)*100, 1)
Bray_PCoA_MATRIX <- Bray_pcoa$vectors[,1:2]
Bray_PCoA_MATRIX <- data.frame(Bray_PCoA_MATRIX)

pc <- c(1,2)
jpeg("Bray-PCoA.png", height = 4, width = 4, units = 'in', res = 600)
plot(Bray_pcoa$vectors[,1:2], bg=c(rgb(154, 0, 0, maxColorValue = 255),rgb(169, 169, 169,
maxColorValue = 255))[Sel_Genes$TYPE], pch=c(21,22)[Sel_Genes$TYPE], cex=1.3,
xlab=paste0("PCo", pc[1], " (", mds.var.per[1], "%)"), ylab=paste0("PCo", pc[2], " (",
mds.var.per[2], "%)"))
ordiellipse(Bray_pcoa$vectors[,1:2], Sel_Genes$TYPE, kind="sd", lwd=1, lty=3, draw =
"polygon", alpha = 50, col = c(rgb(154, 0, 0, maxColorValue = 255),rgb(169, 169, 169,
maxColorValue = 255)))
ordispider(Bray_pcoa$vectors[,1:2], Sel_Genes$TYPE, lty=3, spider ="centroid", lwd=1,
col="black")
legend("top", legend = c("KO", "WT"), col = c(rgb(154, 0, 0, maxColorValue = 255),rgb(169, 169,
169, maxColorValue = 255)),lty = c(1,1,1,1), cex=0.7, title = "", border = "white", fill = NULL,
bg="white", bty = "n")
abline(h=0, v=0, col = "gray60")
dev.off ()

adonis(Bray_distances ~ Sel_Genes$TYPE)
#OUTPUT will be like this (R2=0.99, pval=0.008**)
#Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
#Sel_Genes$TYPE  1   0.39235 0.39235  1554.8 0.99488  0.008 **
```

Code-5 : Volcano plot

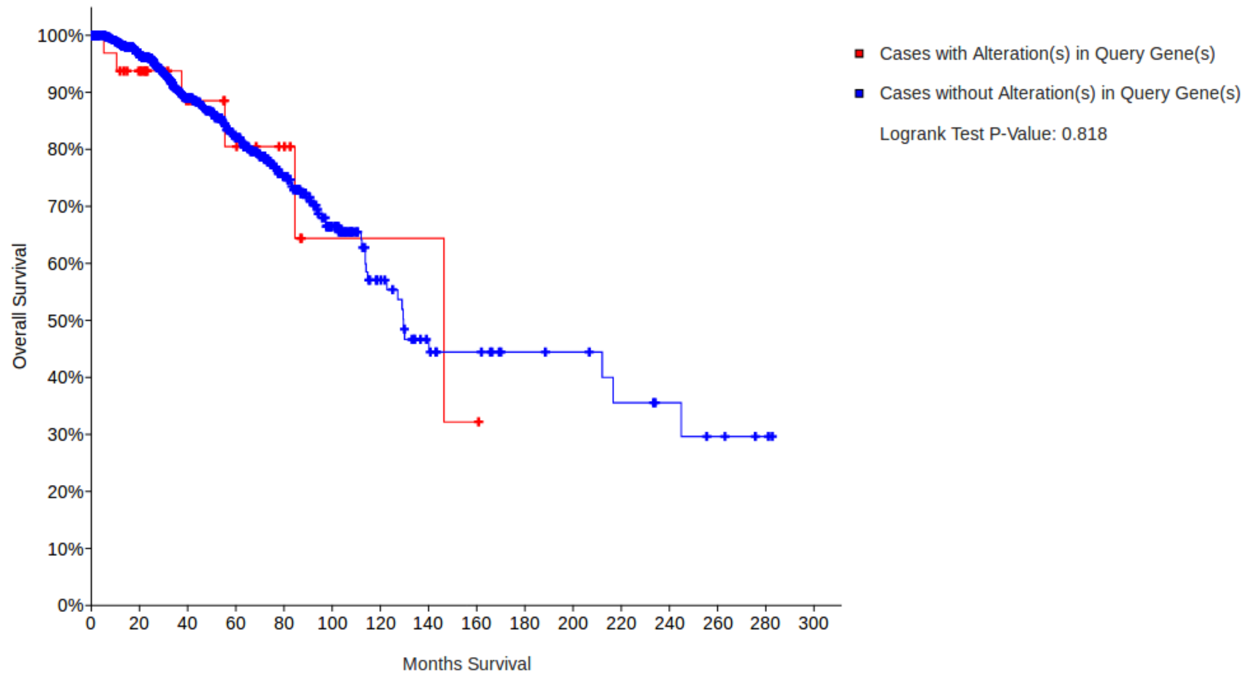
```
###independently from external dataset########
library(EnhancedVolcano)
rpl <- read.table("/home/ravi/Documents/Volcano.R/Data_volcano..csv", header = TRUE, sep =
",")
head(rpl)
lab <- rpl$Gene.symbol
x<- rpl$logFC
res
y<- rpl$P.Value
lab <-rpl$Gene.symbol
EnhancedVolcano(rpl,
            lab = rpl$Gene.symbol,
            x = 'logFC',
            y = 'P.Value',
            selectLab = NULL,
            title = NULL,
            cutoffLineType = 'twodash',
            cutoffLineWidth = 0.8,
            xlim = c(-8,8),
            xlab = bquote(~Log[2]~ 'fold change'),
            ylim = c(0,12),
            ylab = bquote(~-Log[10]~italic(P)),
            pCutoff = 0.05,
            FCcutoff = 1.0,
            transcriptPointSize = 0.5,
            transcriptLabSize = 4.0,
            colAlpha = 1,
            shape = 19,
            subtitle = NULL,
            legend=c('NS','Log (base 2) fold-change','P value',
                   'P value & Log (base 2) fold-change'),
            legendPosition = 'top',
            legendLabSize = 12,
            legendIconSize = 4.0,
            gridlines.major = FALSE,
            gridlines.minor = FALSE,
            drawConnectors = FALSE,
            widthConnectors = 0.2,
            colConnectors = 'grey50',
            border = 'full' )
```

Code-6 : Differential gene expression using DEseq

```
library("DESeq2")
library(ggplot2)
countsName <- "http://bioconnector.org/workshops/data/airway_scaledcounts.csv"
#download.file(countsName, destfile = "airway_scaledcounts.csv", method = "auto")

countData <- read.table(file = "Mnemiopsis_count_data.txt", header = T, sep = "\t")
head(countData)
#metaDataName <- "http://bioconnector.org/workshops/data/airway_metadata.csv"
#download.file(metaDataName, destfile = "airway_metadata.csv", method = "auto")

metaData <- read.table(file = "Mnemiopsis_col_data.txt", header = T, sep = "\t")
metaData
dds <- DESeqDataSetFromMatrix(countData=countData,
                              colData=metaData,
                              design=~condition)
dds
dds <- DESeq(dds)
?DESeq
res <- results(dds)
head(results(dds, tidy=TRUE))
summary(res) #summary of results
res <- res[order(res$padj),]
head(res)
par(mfrow=c(2,3))

plotCounts(dds, gene="ML087114a", intgroup="dex")
plotCounts(dds, gene="ENSG00000179094", intgroup="dex")
plotCounts(dds, gene="ENSG00000116584", intgroup="dex")
plotCounts(dds, gene="ENSG00000189221", intgroup="dex")
plotCounts(dds, gene="ENSG00000120129", intgroup="dex")
plotCounts(dds, gene="ENSG00000148175", intgroup="dex")

par(mfrow=c(1,1))
# Make a basic volcano plot
with(res, plot(log2FoldChange, -log10(pvalue), pch=20, main="Volcano plot", xlim=c(-3,3)))

# Add colored points: blue if padj<0.01, red if log2FC>1 and padj<0.05)
with(subset(res, padj<.01 ), points(log2FoldChange, -log10(pvalue), pch=20, col="blue"))
with(subset(res, padj<.01 & abs(log2FoldChange)>2), points(log2FoldChange, -log10(pvalue),
pch=20, col="red"))
vsdata <- vst(dds, blind=FALSE)
plotPCA(vsdata, intgroup="dex") #using the DESEQ2 plotPCA fxn we can
```

<u>Code 7 : Survival analysis</u>

```
###mydata##########
library(survival)
library(survminer)
getwd()
data <- read.csv("Early_ER_survival_2.csv", header = T, sep = ",")
mySurv <- Surv(time=data$days_to_last_followup, event = data$vital_status)
class (mySurv)
head(mySurv)
myfit <- survfit(mySurv~data$NLGN3)
myfit
median(data$days_to_last_followup)
plot(myfit)
table(data$days_to_last_followup)
plot(myfit, col = c("red", "blue"))
plot(myfit, col = c("red", "blue"),mark=3)
legend("topright", c("no_alteration", "alteration"), col=c("red","blue"),lty=1)
abline(h=0.5)
abline(v=1500)
survdiff(mySurv~data$NLGN3)
#plot(myfit, fun="event")
coxph(mySurv~data$NLGN3)
ggsurvplot(myfit, data = data, pval = TRUE)
```

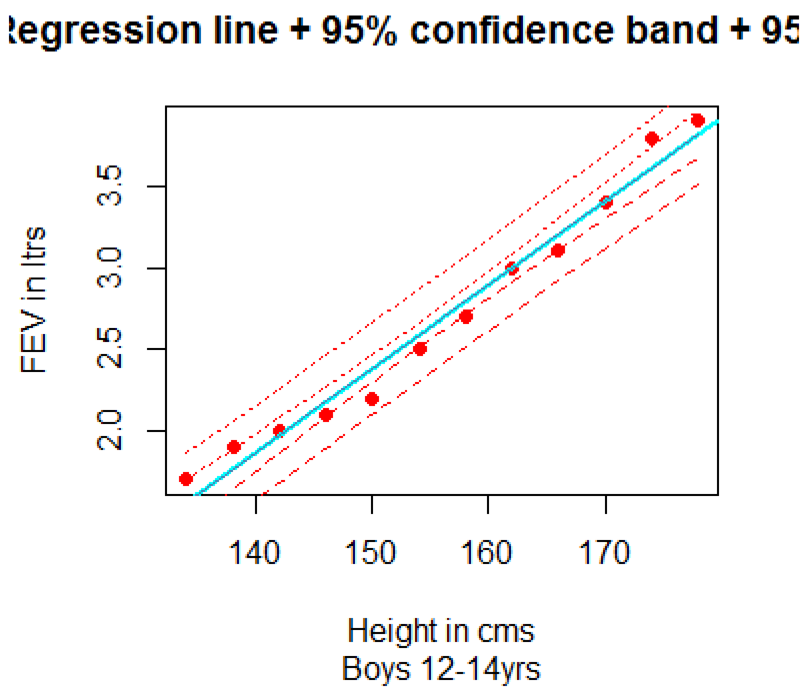


Machine learning models using R

Code 8: Regression analysis

```
Height <- seq(134, 178, 4) #min=134, max= 178 and interval at 4
# or hieght <- seq(134,138,142 and so on)
FEV <- c(1.7, 1.9, 2.0, 2.1, 2.2, 2.5, 2.7, 3.0, 3.1, 3.4, 3.8, 3.9)
MB <- data.frame(Height,FEV)
MB
MB1 <- lm(FEV~Height,data=MB)
summary(MB1)
names(MB1)
round(MB1$fitted.values,2)
MB$Predicted <- round(MB1$fitted.values,2)
MB
MB$Residuals <- round(MB1$residuals, 2)
MB
RSS <- sum((MB1$residuals)^2) ##using residuals, RSS=residual sum of squares
RSS
TSS <- sum ((FEV-mean(FEV))^2)  ###substracting each FEv value from mean to calculate
deviation from mean and then square and sum
TSS
Good <- 1-RSS/TSS
Good
####check normality using residuals
shapiro.test(MB1$residuals) ###null hypothesis is true, p-value is > 5% and data is normally
distributed

####now graphics
plot(MB$Height, MB$FEV, xlab = "Height in cms", ylab = "FEV in ltrs",col="red", pch=16, main =
"scatter plot + Regression line + 95% confidence band + 95% prediction band"
    , sub = "Boys 12-14yrs")
abline(MB1,col="cyan", lwd =2) ###MB1 contains regression data
##calculate 95% confidence band
MB2 <- predict(MB1, newdata = MB, int = "c")
MB2
####matlines(MB2$lwr, MB2$upr, ity = c(1,2,2),col=c("steelblue", "red","red" ))
##so u can plot confidence and prediction bands urself following the tutorial
see <- MB$Height
matlines(see, MB2, lty = c(1,2,2),col=c("steelblue", "red","red" ))
##so u can plot confidence and prediction bands urself following the tutorial
###prediction band
MB3 <- predict(MB1,newdata = MB, int = "p")
matlines(see, MB3, lty=c(1,2,2), col =c("steelblue", "red", "red"))  ####prediction band
```

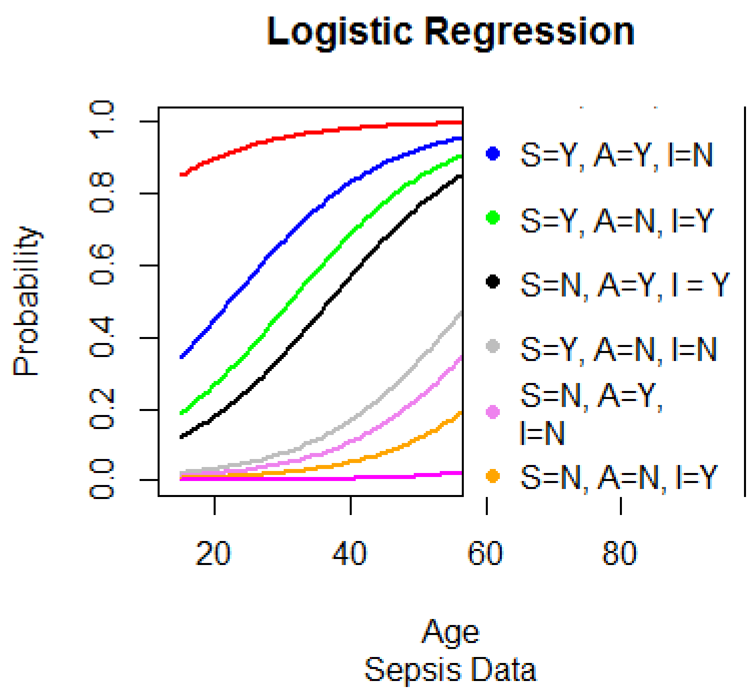
```
####one example data ploting prediction and confidence bands
xv <- seq(130, 180, 0.5)
xv
yv <- predict(MB1, list(Height = xv), int = "c")
(yv)[1:9,]
##head(MB)
matlines(xv,yv,lty=c(1,2,2), col=c("steelblue", "red","red"))
yv1 <- predict(MB1, list(Height = xv), int = "p")
matlines(xv,yv1,lty=c(1,2,2), col=c("steelblue", "red","red"))
```



<u>Code 9 : Logistic regression</u>

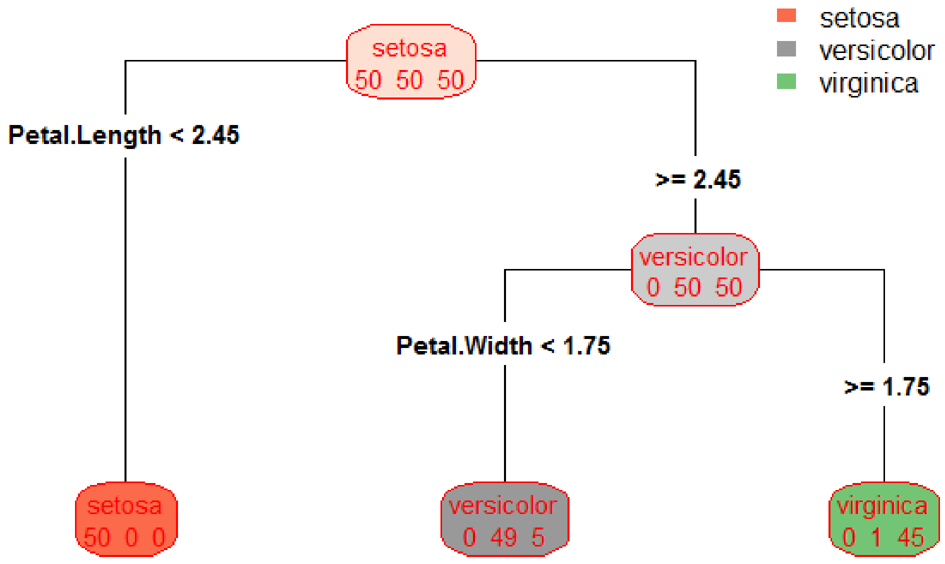
```
##response var is binary, yes or no
##predictor var are binary and 1 is numeric
Sepsis <- read.csv("/Users/apple/Documents/R_workshop_practice/day3/Sepsis.csv")
##Sepsis <- read.delim("clipboard")  ##using R editor
dim(Sepsis)
head(Sepsis)
Sepsis1 <- glm(Outcome~., data = Sepsis, family = binomial) ##glm=generalized linear model,
eg logistic regression, if not given family = binomial, it runs multiple regression.
summary(Sepsis1)
p <- pchisq(53.122,100,lower.tail = FALSE)
p
```

```
data <- data.frame(Shock=c(1,1), Malnutrition=c(0,0), Alcoholism = c(0,0), Age=c(30,70),
Infarction=c(1,1))
data
predict(Sepsis1, newdata = data, type="response") ###calculate chances of death of patient
using above data
Sepsis2 <- glm(Outcome ~ Shock + Alcoholism + Age + Infarction, data = Sepsis, family =
binomial)
summary(Sepsis2)
##calculating goodness of fit
p <- pchisq(56.073,101,lower.tail= FALSE)
p
curve(exp(-8.89458992 + 3.70119321 + 3.18590397 + 0.08983175*x +
        2.38646847)/ (1 + exp(-8.89458992 + 3.70119321 + 3.18590397 + 0.08983175*x +
                        2.38646847)),from = 15, to = 95, lwd = 2, col = "red", ylim = c(0, 1), xlab =
"Age",
    ylab = "Probability", main = "Logistic Regression", sub = "Sepsis Data")

curve(exp(-8.89458992 + 3.70119321 + 3.18590397 + 0.08983175*x)/
      (1 + exp(-8.89458992 + 3.70119321 + 3.18590397 + 0.08983175*x)), col =
      "blue", lwd = 2, add = T)
curve(exp(-8.89458992 + 3.70119321 + 0.08983175*x + 2.38646847)/
      (1 + exp(-8.89458992 + 3.70119321 + 0.08983175*x + 2.38646847)), col =
      "green", lwd = 2, add = T)
curve(exp(-8.89458992 + 3.18590397 + 0.08983175*x + 2.38646847)/
      (1 + exp(-8.89458992 + 3.18590397 + 0.08983175*x + 2.38646847)), col =
      "black", lwd = 2, add = T)
curve(exp(-8.89458992 + 3.70119321 +0.08983175*x)/
      (1 + exp(-8.89458992 + 3.70119321 + 0.08983175*x)), col = "gray", lwd = 2, add
    = T)

curve(exp(-8.89458992 + 3.18590397 + 0.08983175*x)/
      (1 + exp(-8.89458992 + 3.18590397 + 0.08983175*x)), col = "violet", lwd = 2,
    add = T)
curve(exp(-8.89458992 + 0.08983175*x + 2.38646847)/
      (1 + exp(-8.89458992 + 0.08983175*x + 2.38646847)), col = "orange", lwd = 2,
    add = T)

curve(exp(-8.89458992 + 0.08983175*x)/(1 + exp(-8.89458992 +
                              0.08983175*x)), col = "magenta", lwd = 2, add = T)

legend("right", pch = rep(16, 8), col = c("red", "blue", "green", "black",
                           "gray", "violet", "orange", "magenta"), legend = c("S=Y, A=Y, I=Y",
                                                          "S=Y, A=Y, I=N", "S=Y, A=N, I=Y",
"S=N, A=Y, I = Y", "S=Y, A=N, I=N", "S=N, A=Y,
I=N", "S=N, A=N, I=Y", "S=N, A=N, I=N"))

names(Sepsis2)
Sepsis2$coefficients

###calculate the prob of 106 patients in sepsis data

Pred <- predict(Sepsis1, newdata = Sepsis, type = "response")
Pred
round(Pred,2)
##concrete prediction, if prob of death, is greater than 50%, he will die, if it is less than 50%, he
will survive
ConcretePred <- ifelse(Pred <0.50,"Survive","Die")
ConcretePred
Sepsis3 <- data.frame (Sepsis, ConcretePred)
Sepsis3
Sepsis4 <- Sepsis3[,6:7]
Sepsis4
##to calculate the prediction accuracy
table(Sepsis4$Outcome, Sepsis4$ConcretePred)
MisClassificationRate <- 9/106
MisClassificationRate
CorrectClassificationRate <- 97/106
CorrectClassificationRate
```



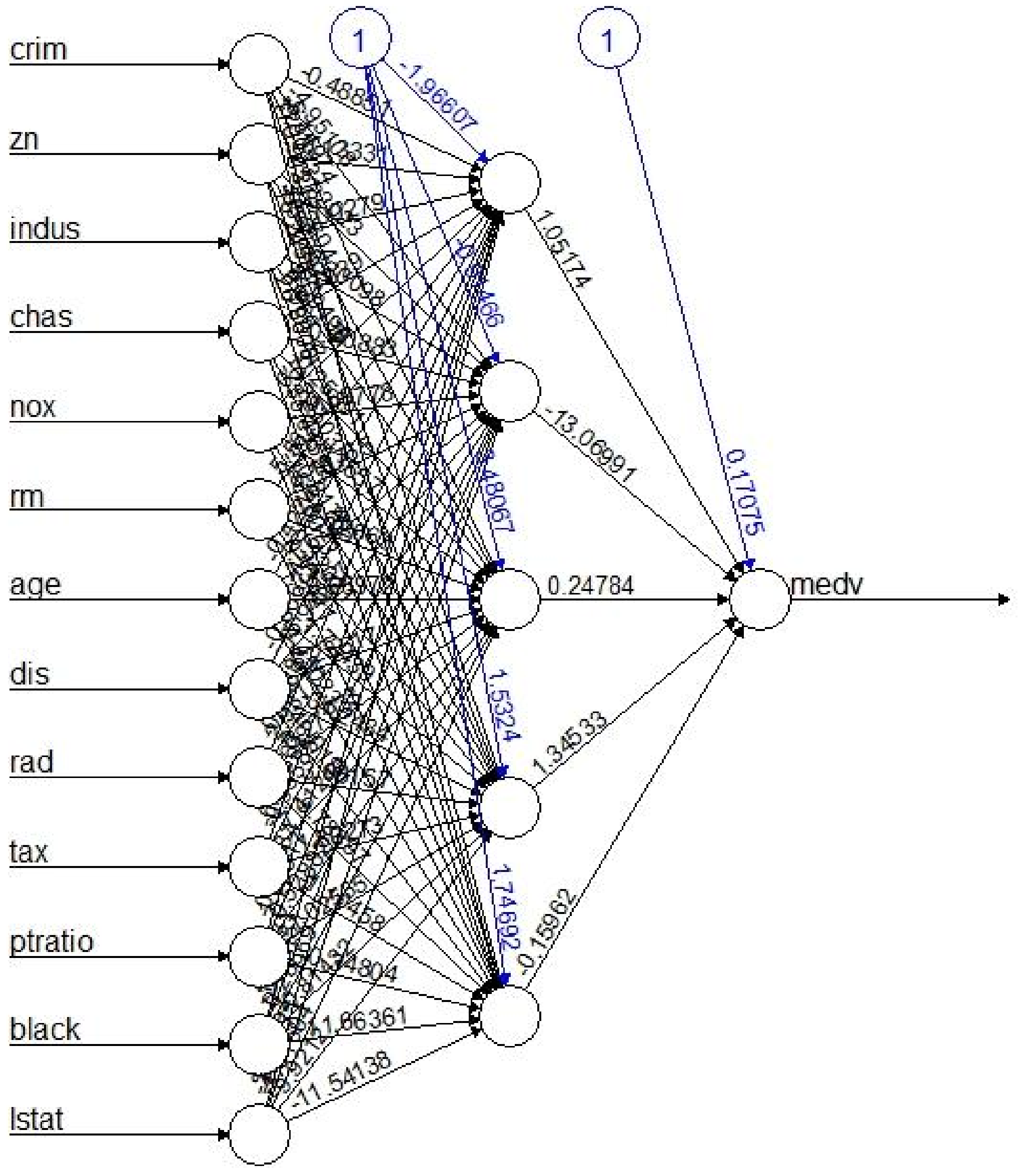
Code 10 : Random Forest

```
library("randomForest")  ###very very efficient package
data("iris")  ###feature of flowers to judge which species it belong to
dim(iris)
head(iris)
###pattern reognition, fisher came up with multivariate analysis, not discussing
summary(iris)
library(party)
MB2 <- randomForest(Species ~ ., data=iris, importance = T) ###importance means feature
selection
print(MB2)
###prediction
Data <- data.frame(Sepal.Length=5.4, Sepal.Width=3.0, Petal.Length=4.7, Petal.Width=1.2)
predict(MB2, Data)
Data
varImpPlot(MB2, pch = 16, col = "red", n.var = 4, sort = T, main =
        "Importance of Variables for the Iris data")
###if u r interested in feature selection, go for random forest.
MB3 <- randomForest(Species ~ Petal.Width + Petal.Length, data=iris, importance = T)
print (MB3)
varImpPlot(MB3, pch = 16, col = "red", n.var = 2, sort = T, main =
        "Importance of Variables from 2 variables")
###difference is very less ,means petal length and width are equally inportant

###Decision tree on iris data
MB4 <- rpart(Species ~ ., data = iris)
rpart.plot(MB4, type=4, extra=1, digits = 3, col = "red")
```


setosa
versicolor
virginica
setosa
50 50 50
Petal.Length < 2.45
>= 2.45
versicolor
0 50 50
Petal.Width < 1.75
>= 1.75
setosa
50 0 0
versicolor
0 49 5
virginica
0 1 45

#Code 11 : Neural networks

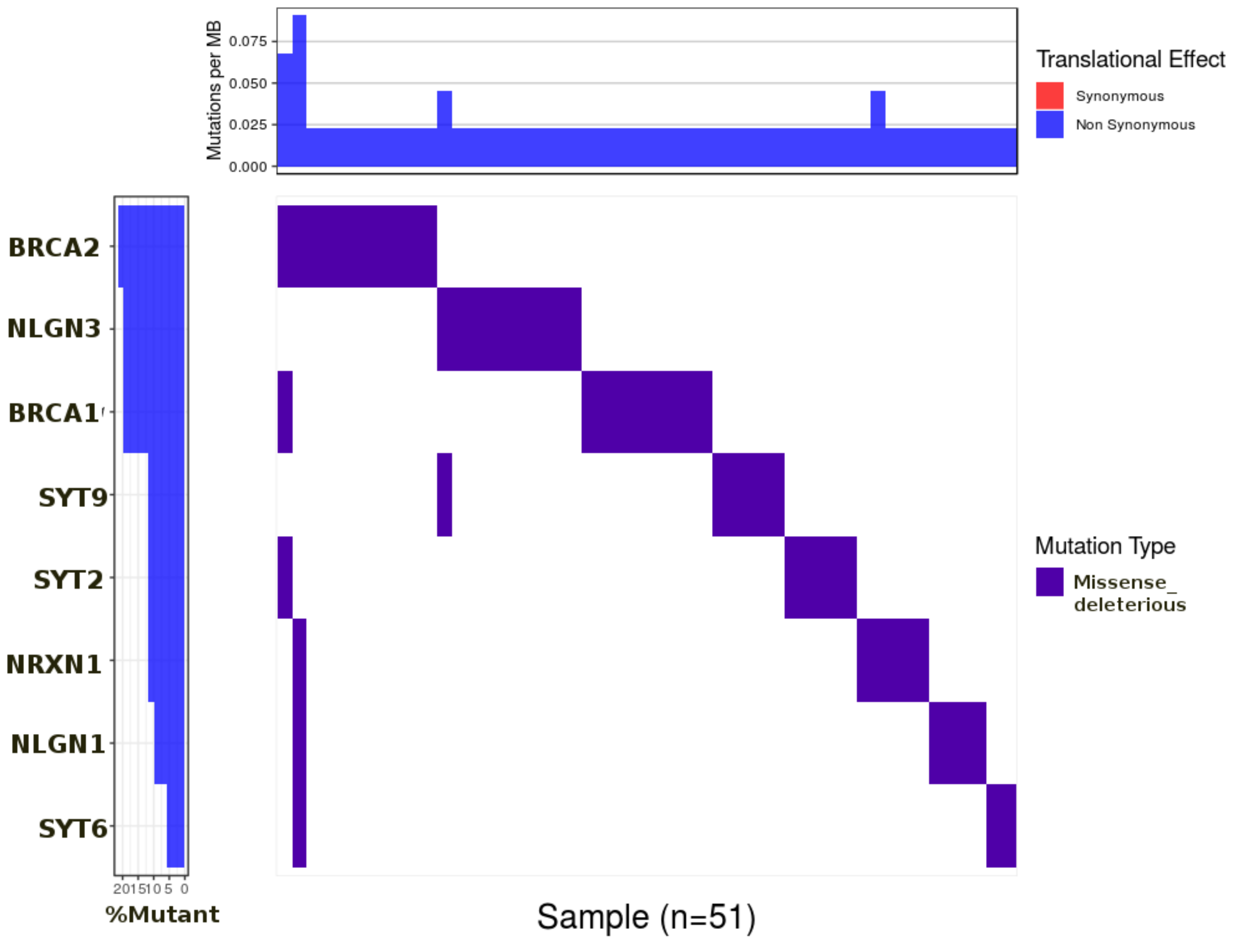
```
data(Boston)
library("MASS")
dim(Boston)
head(Boston)
###Multiple regression model
Rao <- lm(medv ~ ., data=Boston)
summary(Rao)
###calculate MSE(mean square error)
names(Rao)
Rao$fitted.values
MSE <- sum( (Boston$medv-Rao$fitted.values)^2)/506
MSE
Maxs<- apply(Boston,2,max)
Mins <- apply(Boston,2,min)
class(Boston)
ScaledB <- scale(Boston, center=Mins,scale=Maxs-Mins)
head(ScaledB)
install.packages("neuralnet")
library("neuralnet")
index <- sample(1:nrow(Boston), round(0.75*nrow(Boston)))
head(index)
train <- ScaledB[index, ]
test <- ScaledB[-index, ]
Neural <- neuralnet(medv ~ crim + zn + indus + chas + nox + rm + age + dis +
                    rad + tax + ptratio + black + lstat, data = train, hidden = 5, linear.output =
                    TRUE)
Neural
plot(Neural)
```



Code 12: Waterfall plot for mutation in R

```
library ("GenVisR")
my_data=read.table(file="/home/ravi/Documents/cbio_oncoprint_data/using_R_package/waterfa
ll/cluster1_deleterious.csv", header=TRUE, sep="\t")
my_data=read.table(file="/home/ravi/Documents/cbio_oncoprint_data/using_R_package/waterfa
ll/cluster_2_deleterious.csv", header=TRUE, sep="\t")
my_data=read.table(file="/home/ravi/Documents/cbio_oncoprint_data/using_R_package/waterfa
ll/cluster_3_deleterious_curated.csv", header=TRUE, sep="\t")
my_data=read.table(file="/home/ravi/Documents/cbio_oncoprint_data/using_R_package/waterfa
ll/cluster_4_deleterious_curated.csv", header=TRUE, sep="\t")
```

```
colnames(my_data)[c(1,2,3)]=c("sample","gene","variant_class")
most_deleterious <- c("missense_deleterious")
waterfall(my_data, fileType = "Custom", variant_class_order = most_deleterious,
mainLabelCol="amino.acid.change", mainLabelSize=3,
        mainLabelAngle = 90)
###removing label
waterfall(my_data, fileType = "Custom", variant_class_order = most_deleterious,
main_geneLabSize =9.0,mainLabelSize=25,
        mainLabelAngle = 90)
#save as ratio 1500:900 and 2000:1100
```



Code 13: Lollipop plot in R

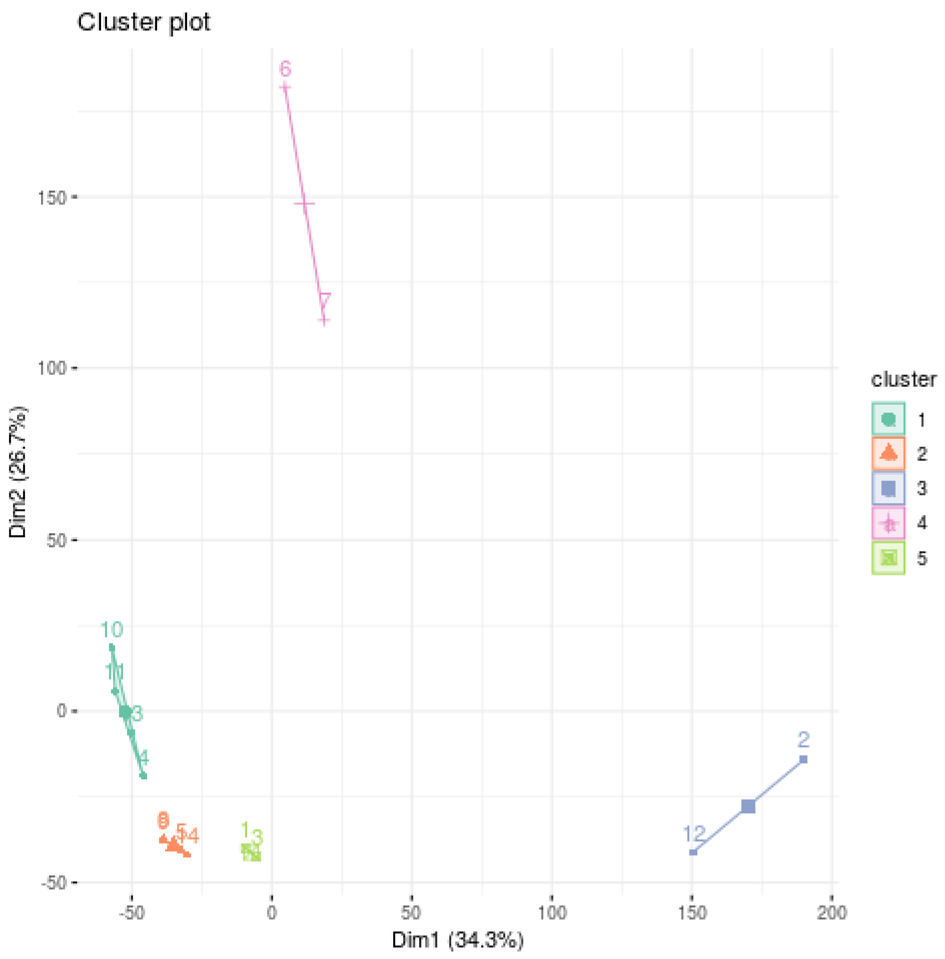
```
library(GenVisR)
##datacuration##
####put "p." prior to amino acid change in input eg G27472R is changed to
p.G27472R#########
```

```
mutation_data=read.table(file="/home/ravi/Documents/cbio_oncoprint_data/using_R_package/lo
lli_R/Maml2_genvisR.csv", header=TRUE, sep="\t")
colnames(mutation_data)[c(1,2,3,4)]=c("gene","amino_acid_change","transcript_name","mutatio
n_type")
MAML2_data=mutation_data[mutation_data$gene=="MAML2",]
lolliplot(MAML2_data, fillCol="NULL", labelCol="amino_acid_change", txtSize=3, txtAngle=20,
host="jan2019.archive.ensembl.org")
lolliplot(MAML2_data, fillCol= "NULL", labelCol="amino_acid_change", txtSize=3, txtAngle=20,
host="oct2018.archive.ensembl.org")
or
library(GenVisR)
mutation_data=read.table(file="/home/ravi/Documents/cbio_oncoprint_data/using_R_package/lo
lli_R/Maml2_genvisR.csv", header=TRUE, sep="\t")
colnames(mutation_data)[c(1,2,3,4)]=c("gene","amino_acid_change","transcript_name","mutatio
n_type")
par(bg = 'white')
lolliplot(mutation_data, fillCol="mutation_type", labelCol="amino_acid_change", proteinColour =
"#999999",
        txtSize=6, pntSize=5, txtAngle=20, host="oct2018.archive.ensembl.org")
#####removing label########
lolliplot(mutation_data, fillCol="mutation_type", proteinColour = "#999999",
        txtSize=6, pntSize=5, txtAngle=80, host="oct2018.archive.ensembl.org")
```

Code 14: K-means clustering in R

```
#Optimization of number of clusters using three methods
#####ELBOW METHOD##########

set.seed(13)
wss <- (nrow(df)-1)*sum(apply(df,2,var))
for (i in 2:12) wss[i] <- sum(kmeans(df,
                                     centers=i)$withinss)
plot(1:12, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")

##printing to file###
set.seed(13)
wss <- (nrow(df)-1)*sum(apply(df,2,var))
for (i in 2:12) wss[i] <- sum(kmeans(df,
                                     centers=i)$withinss)
jpeg (file = paste(wss[i], '.jpg'))
plot(1:12, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
dev.off()

##Method2:SILHOUETTE ANALYSIS##

library(fpc)
set.seed(13)
pamk.best2 <- pamk(df)
cat("number of clusters estimated by optimum average silhouette width:", pamk.best2$nc, "\n")

number of clusters estimated by optimum average silhouette width: 5

##Method2:GAP STATISTICS##

library(cluster)
set.seed(13)
clusGap(df, kmeans, 10, B = 100, verbose = interactive())
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 100)  [one "." per sample]:
 .................................................. 50
.................................................. 100
Clustering Gap statistic ["clusGap"] from call:
 clusGap(x = df, FUNcluster = kmeans, K.max = 10, B = 100, verbose = interactive())
B=100 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
--> Number of clusters (method 'firstSEmax', SE.factor=1): 5
```

```
results <- kmeans(df,5)
results$cluster
results$size
table(data$Gene_name,results$cluster)
library("factoextra")
fviz_cluster(results, data = df)
```


Cluster plot
Dim1 (34.3%)
Dim2 (26.7%)
cluster